Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY


Instytut Elektrotechniki Teoretycznej i Systemów Informacyjno-Pomiarowych

# Praca dyplomowa magisterska

na kierunku Informatyka Stosowana

w specjalności Inżynieria Oprogramowania

Porównanie efektywności wybranych narzędzi służących do serwowania danych

inż. Jan Łukomski

numer albumu 291089

promotor

prof. dr hab. inż. Remigiusz Rak

WARSZAWA 2024

**Porównanie efektywności wybranych narzędzi służących do serwowania danych**

**Streszczenie**

TODO na całą stronę

**Słowa kluczowe:** A, B, C

## Comparison of the effectiveness of selected data serving tools

**Abstract**

TODO

**Keywords:** X, Y, Z

# Spis treści

# Rozdział 1

# Wstęp

## 1.1 Cel pracy i zakres pracy

# Rozdział 2

# Przegląd istniejących badań dla wybranych narzędzi

# Rozdział 3

# Opis koncepcji badania

## 3.1 Badanie pojedyńczego zapytania

## 3.2 Badania limitu użytkowników

# Rozdział 4

# Opis wykorzystywanych narzędzi i bibliotek

## 4.1 Django

Django to framework do tworzenia aplikacji insternetowych [3]. Jest to narzędzie, które nadaje się do szybkiego tworzenia zaawansowanych aplikacji internetowych, zachowując jednocześnie wysoką jakość i bezpieczeństwo kodu. Jego popularność wynika z wielu czynników, w tym szybkości wdrażania projektów, bogatej palety wbudowanych funkcji oraz zabezpieczeń.

Jedną z głównych zalet Django jest jego zdolność do szybkiego rozwoju aplikacji od pomysłu do wdrożenia. Framework ten oferuje wiele narzędzi i gotowych rozwiązań, które ułatwiają proces tworzenia stron internetowych, od autentykacji użytkowników po obsługę treści czy generowanie map strony. Dzięki temu programiści mogą skupić się na kształtowaniu logiki aplikacji. Posiada on wbudowane moduły, które pozwalają na szybki start projektu. Są to mechanizmy obrony przed najczęstszymi atakami, takimi jak wstrzykiwanie SQL czy ataki typu cross-site scripting, czy system autentykacji użytkowników, zarządzanie kontami użytkowników oraz hasłami.

Django pozwala aplikacjom na elastyczne dostosowywanie się do zmieniających się wymagań i wzrostu liczby użytkowników. Może być stosowany zarówno w małych projektach, jak i w dużych systemach obsługujących ogromne ruchy internetowe, co czyni go uniwersalnym narzędziem dla różnorodnych zastosowań.

Kod źródłowy frameworka Django jest udostępniony publicznie na zasadach otwartego oprogramowania.

## 4.2 .NET

.NET Framework, opracowany przez firmę Microsoft, stanowi kompleksową platformę programistyczną, która umożliwia tworzenie różnorodnych aplikacji komputerowych [1]. Jest to zintegrowane środowisko programistyczne zawierające narzędzia, biblioteki oraz środowisko wykonawcze, które

ułatwiają proces tworzenia oprogramowania. Zapewnia infrastrukturę do uruchamiania, rozwijania i zarządzania aplikacjami na platformie Windows oraz innych systemach operacyjnych.

Jedną z charakterystycznych cech .NET Framework jest jego wieloplatformowość, co oznacza możliwość pisania kodu raz i uruchamiania go na różnych systemach operacyjnych, takich jak Windows, Linux czy macOS. Taka elastyczność sprawia, że framework ten jest atrakcyjny dla programistów oraz firm poszukujących rozwiązania umożliwiającego tworzenie aplikacji na różnych platformach. W skład .NET Framework wchodzi rozbudowany zestaw bibliotek i narzędzi, które usprawniają proces tworzenia oprogramowania. Obejmuje on między innymi biblioteki do obsługi interfejsu użytkownika, zarządzania pamięcią, komunikacji sieciowej oraz dostępu do danych.

.NET Framework jest zintegrowy innymi technologiami Microsoftu, takimi jak Visual Studio czy platforma chmurowa Azure. Umożliwia to programistom korzystanie z kompleksowych narzędzi do tworzenia, testowania i wdrażania aplikacji, a także skalowania ich w chmurze.

## 4.3 NestJS

NesJS jest frameworkiem dedykowanym do tworzenia wydajnych, skalowalnych aplikacji serwerowych opartych na Node.js [4]. Jest oparty na progresywnym JavaScriptie, wspiera w pełni TypeScript (pozwalając jednocześnie programistom pisać w czystym JavaScriptie) oraz łączy elementy programowania obiektowego (OOP), programowania funkcyjnego (FP) i programowania funkcyjnego reaktywnego (FRP).

Framework Nest wykorzystuje renomowane platformy HTTP Server, takie jak Express (domyślnie) i opcjonalnie Fastify. Daje to programistom swobodę wyboru między nimi, jednocześnie oferując abstrakcję ponad nimi oraz dostęp do ich interfejsów API.

Filozofia frameworka Nest opiera się na potrzebie stworzenia spójnej architektury aplikacji, które umożliwiają łatwe testowanie, skalowalność, luźne powiązania oraz łatwe utrzymanie kodu. Inspiracją dla tej architektury były koncepcje stosowane w frameworku Angular.

Nest jest projektem o otwartym źródle na licencji MIT, rozwijanym od 2017 roku.

## 4.4 K6

Grafana k6 to narzędzie do testowania obciążenia aplikacji internetowych oraz wykonywania testów wydajnościowych [2]. Jest to część ekosystemu Grafana, znanej platformy do monitorowania i analizy danych, co zapewnia użytkownikom możliwość integracji testów wydajnościowych z analizą danych i wizualizacją wyników.

Jedną z kluczowych cech narzędzia Grafana k6 jest jego zdolność do symulowania zachowania użytkowników poprzez wysyłanie zapytań HTTP i analizowanie odpowiedzi serwera. Można tworzyć zaawansowane scenariusze testowe, które odwzorowują różne zachowania użytkowników na stronie internetowej, takie jak logowanie, przeglądanie stron, czy też dodawanie produktów do koszyka. Oferuje ono również bogate możliwości konfiguracyjne, które pozwalają dostosować testy do różnych

scenariuszy. Można określić warunki obciążeniowe, definiować progi wydajnościowe oraz zbierać szczegółowe dane diagnostyczne, które pomagają zidentyfikować przyczyny ewentualnych problemów.

Kod źródłowy k6 jest dostępny publicznie, co oznacza, że jest dostępny dla szerokiej społeczności deweloperów i testerów. Dzięki temu można korzystać z bogatej dokumentacji, zgłaszać błędy oraz współpracować nad rozwojem narzędzia w ramach społeczności.

Grafana k6 to wszechstronne i potężne narzędzie do testowania wydajności aplikacji internetowych, które pozwala użytkownikom na symulowanie różnych scenariuszy obciążeniowych oraz monitorowanie wydajności. Dzięki temu deweloperzy i testerzy mogą zapewnić, że ich aplikacje są wydajne i odpowiadają na oczekiwania użytkowników.

## 4.5 Docker

# Rozdział 5

# Przygotowanie aplikacji

## 5.1 Zbiory danych

W celu przeprowadzenia badania konieczne było przygotowanie odpowiednich zbiorów danych, które miały posłużyć do symulacji różnych scenariuszy. W tym kontekście przygotowano dwa zbiory danych, aby umożliwić różnorodne analizy:

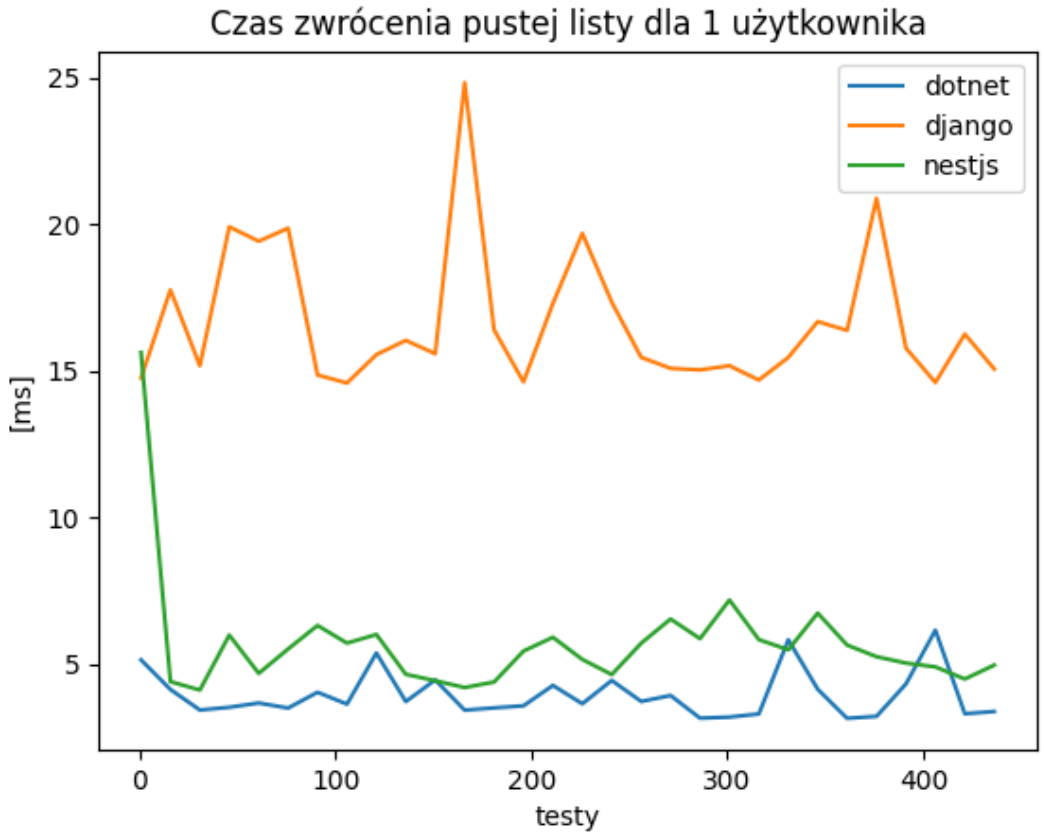
- **FWB_0** - Jest to zbiór pusty, pozbawiony jakichkolwiek elementów. Brak danych w tym zbiorze ma posłużyć do sprawdzenia zachowania systemu w sytuacji, gdy nie ma żadnych rekordów do przetworzenia.
- **FWB_100K** - Ten zbiór składa się z 100 000 elementów. Każdy element tego zbioru reprezentuje pojedynczy rekord w bazie danych i zawiera unikalne identyfikatory (numery) oraz nazwy (tekstowe). Zbiór ten został przygotowany w celu przetestowania wydajności systemu oraz jego reakcji na duże ilości danych.

Jeden element to rekord w bazie danych zawierający ID (number) oraz nazwę (tekst). Przygotowanie tych zbiorów danych stanowiło niezbędny krok przed przystąpieniem do właściwej analizy i symulacji różnych scenariuszy w badaniu. Dzięki tym zbiorom możliwe było zbadanie zachowania systemu w różnych warunkach oraz przeprowadzenie odpowiednich wniosków na podstawie uzyskanych wyników.

# Rozdział 6

# Badanie aplikacji

## 6.1 Pojedyńcze zapytania




**Rysunek 1.** Czas zwrócenia pustej listy dla 1 użytkownika

Zdecydowanym liderem zestawienia jest Dotnet

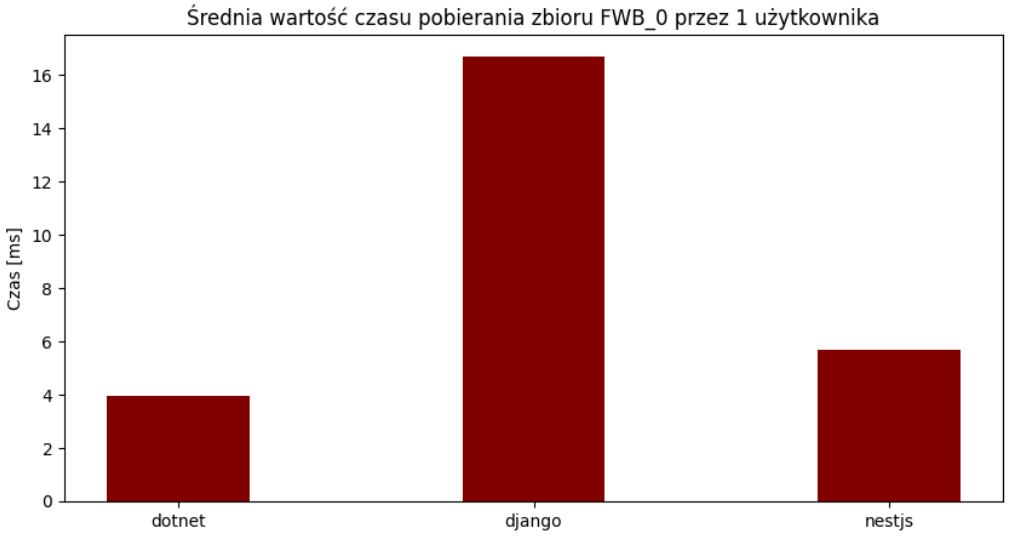
## 6.2 Limity równoległych zapytań

| Narzędzie | Liczba użytkowników (**FWB_0**) | Liczba użytkowników (**FWB_100K**) |
|---|---|---|
| Django | 3159 | 3070 |
| Dotnet | 14469 | 90 |
| NestJS | 4132 | 0 |

Django potrafi obsługiwać podobną ilość użytkowników dla obu zbiorów. W przypadku zbioru danych FWB_100K, narzędzie zanotowało spadek o niecałe 3%.

Dotnet jest zdecydowanym liderem liczby użytkowników dla pustego zbioru. Widać u niego jednak znaczny spadek liczby obsługiwanych użytkowników wraz ze wzrostem ilości przesyłanych danych.

NestJS które przy zbiorze pustym jest nieco lepsze od Django, dla zbioru FWB_100K całkowicie przestało odpowiadać - co oznacza, że nie radzi sobie z taką ilością danych.




**Rysunek 2.** Średni czas zwrócenia pustej listy dla 1 użytkownika

# Rozdział 7

# Podsumowanie i wnioski

# Bibliografia


[1] *.NET | Build. Test. Deploy. — dotnet.microsoft.com*, https://dotnet.microsoft.com/, [Accessed 17-03-2024].

[2] *Browser Module Documentation — k6.io*, https://k6.io/docs/using-k6-browser/overview/, [Accessed 17-03-2024].

[3] *Django overview — djangoproject.com*, https://www.djangoproject.com/start/overview/, [Accessed 17-03-2024].

[4] *Documentation | NestJS - A progressive Node.js framework — docs.nestjs.com*, https://docs.nestjs.com/, [Accessed 17-03-2024].

# Wykaz skrótów i symboli

# Spis rysunków

# Spis tablic

# Spis załączników

# Załącznik 1
# Dowód próżni doskonałej

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

# Załącznik 2
# Dowód zera bezwzględnego

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

# Załącznik 3

# Dowód czasu zatrzymanego

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

# Załącznik 4
# Dowód nieskończoności urojonej

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.